BIOF3 组学数据分析

基础入门手册

BioF3 基础入门专题导出版

导出日期：2026年5月11日

扫码关注微信公众号【生信F3】

获取文章完整内容，分享生物信息学最新知识。

BIOF3 组学数据分析

基础入门手册目录

BioF3 基础入门专题导出版

01 组学数据分析入门

欢迎来到 BioF3 组学数据分析教程。如果你是第一次接触生物信息学，或者想系统学习组学数据分析，这篇文章会帮你建立一个清晰的认知框架。

从一个问题开始

假设你手上有一份单细胞 RNA 测序数据，包含 10000 个细胞，每个细胞测到了 20000 个基因的表达量。这是一个 10000 × 20000 的矩阵，200 万个数据点。

你想回答的问题可能是：

- 这些细胞分成几种类型？
- 不同细胞类型的标志基因是什么？
- 细胞之间如何相互作用？
- 某个基因在哪些细胞中高表达？

这就是组学数据分析要解决的核心问题：**从海量数据中提取生物学意义**。

组学数据的特点

数据量大

一次单细胞实验可能产生几十 GB 的原始数据。传统的 Excel 打开都会卡死，必须用编程的方式处理。

维度高

基因组有 3 万个基因，转录组可能检测 2 万个基因，蛋白质组有上万个蛋白。这种高维数据需要降维才能可视化和理解。

噪音多

生物学实验本身就有变异，测序技术也有误差。数据中混杂着真实的生物学信号和技术噪音，需要统计方法来区分。

稀疏性

单细胞数据中，很多基因在某个细胞中表达量为 0。这种稀疏性是技术限制，也是生物学真实情况的反映。

分析流程概览

组学数据分析通常包含以下几个步骤：

1. 数据获取

从测序仪下机的是原始数据（FASTQ 格式），包含每条 reads 的序列和质量值。这一步需要了解测序原理和数据格式。

2. 质量控制

检查测序质量，过滤低质量数据。这一步决定了后续分析的可靠性。常用工具有 FastQC、MultiQC。

3. 序列比对

将 reads 比对到参考基因组，确定每条 reads 来自哪个基因。单细胞数据常用 Cell Ranger、STARsolo 等工具。

4. 表达矩阵构建

统计每个基因在每个细胞中的表达量，得到基因表达矩阵。这是后续分析的起点。

5. 数据标准化

不同细胞的测序深度不同，需要标准化才能比较。常用方法有 CPM、TPM、SCTransform 等。

6. 降维和聚类

用 PCA、UMAP、t-SNE 等方法将高维数据投影到 2D 或 3D 空间，然后用聚类算法识别细胞类型。

7. 差异分析

比较不同组之间的基因表达差异，找到标志基因或差异表达基因。

8. 功能注释

对差异基因进行 GO、KEGG 等功能富集分析，理解生物学意义。

9. 可视化

用热图、小提琴图、散点图等方式展示结果。好的可视化能让复杂的数据一目了然。

需要掌握的技能

编程能力

R 语言是单细胞分析的主流工具，Seurat、Scanpy 等软件包都有 R 版本。R 的优势在于统计分析和可视化。

Python 在数据处理和机器学习方面更灵活，Scanpy、AnnData 是 Python 生态的代表。

Linux 命令行用于服务器操作和流程管理。很多生物信息学工具只有命令行版本。

你不需要成为编程专家，但要能看懂代码、修改参数、调试错误。

AI 辅助能力

AI 编程工具可以帮助你解释代码、生成脚本草稿、定位报错、整理项目文档。但在组学分析中，AI 不能替你判断实验设计、统计方法和生物学结论。把 AI 当作助教和开发助手，让它提高效率，关键结果自己把关。详见 AI 辅助编程与智能体工具。

统计学基础

组学数据分析本质上是统计问题。你需要理解：

- 什么是 p 值和 FDR（错误发现率）
- 为什么要做多重检验校正
- 如何选择合适的统计检验方法
- 什么是过拟合和欠拟合

生物学背景

数据分析的目的是回答生物学问题。你需要知道：

- 基因表达的调控机制
- 细胞类型的标志基因
- 信号通路和代谢通路
- 疾病相关的生物学过程

不需要成为生物学专家，但要能读懂文献，理解实验设计。

常用工具和平台

单细胞分析

- **Seurat**（R）：最流行的单细胞分析工具，功能全面
- **Scanpy**（Python）：Python 版的 Seurat，速度更快
- **Cell Ranger**：10x Genomics 官方的数据处理流程
- **Monocle**：轨迹推断和拟时序分析
- **CELLxGENE / HCA / GEO / SRA**：公开数据检索、浏览和下载

数据可视化

- **ggplot2**（R）：强大的绘图系统，语法优雅
- **matplotlib/seaborn**（Python）：Python 的绘图库
- **Plotly**：交互式可视化
- **IGV**：基因组浏览器

计算环境

- **Jupyter Notebook**：交互式编程环境，适合探索性分析
- **RStudio**：R 语言的集成开发环境
- **Docker**：容器化技术，保证环境一致性
- **Conda**：包管理工具，简化软件安装
- **Codex / Claude Code / opencode / Kiro**：AI 辅助编程和项目维护工具

学习路径建议

第一阶段：打基础（2-4 周）

先学会一门编程语言（R 或 Python），能写简单的脚本，处理数据文件。同时了解 Linux 基本命令。

推荐从本教程的编程基础开始。

在开始之前，强烈建议先看一下AI 辅助编程与智能体工具，建立使用 AI 的边界和姿态，后续每一步都会更稳。

这一阶段的目标不是“学完编程”，而是能完成一个最小可复现的小任务：

- 读取一张表
- 做一次筛选
- 画一张图
- 保存结果
- 记录版本和参数

第二阶段：跑通流程（4-8 周）

跟着教程完整跑一遍单细胞分析流程，从原始数据到最终结果。不要求理解每个参数，先把流程跑通。

推荐学习单细胞实践 01 到04。

第三阶段：深入理解（2-3 个月）

回过头来理解每一步的原理，尝试调整参数，看看结果如何变化。阅读相关文献和软件文档。

推荐学习单细胞实践 05 到12。

第四阶段：独立分析（持续）

用自己的数据或公开数据集做完整分析，遇到问题查文档、搜索、提问。逐渐形成自己的分析思路。

常见误区

误区 1：工具越新越好

新工具可能有新功能，但也可能不稳定。成熟的工具（如 Seurat、DESeq2）经过大量验证，更可靠。

误区 2：参数越复杂越好

很多默认参数已经经过优化。盲目调参可能引入偏差。理解参数含义比调参更重要。

误区 3：p 值越小越好

$p < 0.05$ 只是一个阈值，不代表生物学意义。要结合 fold change、生物学背景综合判断。

误区 4：可视化越炫越好

清晰准确比炫酷重要。简单的散点图、柱状图往往比复杂的 3D 图更有效。

实用建议

记录分析过程

用 Jupyter Notebook 或 R Markdown 记录每一步操作和参数。几个月后你会忘记当时为什么这么做。

版本控制

用 Git 管理代码和分析脚本。这样可以追溯历史版本，也方便团队协作。

数据备份

原始数据和中间结果要备份。硬盘损坏、误删除的情况时有发生。

多看文献

看看别人怎么分析类似的数据，学习他们的思路和方法。Nature、Cell、Science 上的单细胞文章都值得精读。

参与社区

加入 Biostars、SEQanswers、生信技能树等社区，提问和回答问题。教别人是最好的学习方式。

下一步

现在你对组学数据分析有了整体认识，可以开始学习具体的技能了：

- 编程基础 - 学习 R 和 Python
- AI 辅助编程与智能体工具
- Jupyter 与交互式分析环境
- 公共数据库与数据检索
- R 数据整理与 ggplot2 可视化
- 单细胞实践 01 - 实践数据集与数据获取

参考资源

- **Seurat 官方教程**：https://satijalab.org/seurat/

- **Scanpy 文档**：https://scanpy.readthedocs.io/
- **单细胞最佳实践**：https://www.sc-best-practices.org/
- **生信技能树**：http://www.biotrainee.com/

这篇文章只是一个起点。组学数据分析是一个需要持续学习的领域，新方法、新工具层出不穷。保持好奇心，多动手实践，你会逐渐建立起自己的分析体系。

02 AI 辅助编程与智能体工具

AI 编程工具已经从“补全几行代码”发展到“阅读项目、修改文件、运行命令、检查结果”的智能体工作流。对组学数据分析来说，它们可以显著提高效率，但不能替代你对数据、统计方法和生物学问题的判断。

本章的目标不是追工具热点，而是建立一套可靠的使用框架：什么时候用 AI，怎么给任务，如何检查结果，哪些数据不能交给外部服务。

什么是 vibe coding

Vibe coding 指用自然语言描述目标，让 AI 快速生成原型代码或应用。它适合探索想法，例如：

- 快速画一张表达量分布图
- 把一段 R 代码改写成 Python
- 根据报错信息定位可能原因
- 生成一个小型数据清洗脚本
- 搭建一个教程页面或分析报告模板

它的问题也很明显：模型可能默认很多假设，生成的代码看起来能跑，但不一定统计上正确、可重复或适合真实数据。

因此在 BioF3 的语境里，更推荐把 vibe coding 当作“快速草稿”，后续必须补上：

- 明确输入和输出
- 固定软件版本
- 保存参数和随机种子
- 人工检查统计方法
- 用小数据集验证结果
- 把一次性代码整理成可复现脚本

智能体工具和普通聊天的区别

普通聊天工具通常只回答问题。智能体工具可以连接到你的代码环境，执行更完整的开发流程：

- 读取项目文件
- 修改多个文件
- 运行终端命令
- 执行测试或构建
- 根据错误继续修复
- 生成提交说明或 Pull Request

这很适合软件开发，也适合整理组学分析流程。但权限越大，风险也越高。第一次使用时，建议让工具只读项目，先让它解释结构和提出计划，再允许它修改文件。

常见工具类型

1. IDE 内置助手

这类工具嵌入编辑器，适合日常写代码、解释局部文件、生成函数和查看 diff。

常见形态包括：

- VS Code 插件
- Cursor 等 VS Code 系编辑器

- JetBrains 插件
- 云端 IDE 或浏览器工作区

适合任务：

- 解释当前脚本
- 补全函数
- 重构局部代码
- 修复语法错误
- 根据已有风格写一段相似代码

不适合任务：

- 没有上下文的大型分析决策
- 未经确认就批量改动整个项目
- 直接处理敏感临床数据

2. Codex

Codex 是 OpenAI 的编码智能体，可以在终端、本地开发环境或相关产品界面中使用。它适合读项目、改文件、运行命令、做代码审查和处理多文件任务。

典型用法：

```
codex
```

可以让它做：

```
梳理这个 Docusaurus 项目的目录结构，指出教程内容、静态资源和部署脚本分别在哪里。
```

或者：

```
检查 docs/modules/module02.md 里的 R 代码块，找出可能无法直接运行的地方，只给建议，不要修改文件。
```

使用建议：

- 先让 Codex 读项目并给计划
- 修改前确认影响范围
- 每次只给一个明确任务
- 改完后运行 `npm run build` 、测试或示例脚本
- 不要把未脱敏数据、密钥、服务器凭据写进提示词

3. Claude Code

Claude Code 是 Anthropic 的编码智能体，可以在终端、IDE、桌面应用和浏览器中使用。它能读代码库、编辑文件、运行命令，并和开发工具集成。

典型用法：

```
claude
```

适合任务：

- 解释陌生代码库

- 根据报错追踪问题
- 编写测试
- 批量修复 lint 或格式问题
- 整理项目文档
- 通过 `CLAUDE.md` 固定项目规则

在生信项目中，可以让它做：

```
阅读这个 R 脚本，解释每一步在单细胞分析流程中的作用，并指出哪些参数需要根据数据集调整。
```

4. opencode

opencode 是开源的 AI 编码智能体，主打终端工作流，也提供桌面和 IDE 形态。它可以连接不同模型供应商，适合希望掌控工具栈和模型来源的开发者。

典型用法：

```
opencode
```

适合任务：

- 本地项目问答
- 生成修改计划
- 实施局部功能
- 维护项目级 `AGENTS.md`
- 在多个模型之间切换

使用 opencode 时要特别注意 API key 管理。不要把 key 写进仓库，不要提交 `.env` 文件。

5. Kiro

Kiro 是偏“规格驱动开发”的 AI IDE。它强调先把需求、设计和任务拆清楚，再让智能体执行。相比纯 vibe coding，Kiro 更适合把原型推进到可维护项目。

它的核心概念包括：

- specs：把需求、设计和任务写成结构化文档
- steering：给项目提供长期规则和上下文
- hooks：在保存、创建或删除文件时触发自动化任务
- agentic chat：通过自然语言和项目交互

适合任务：

- 从想法生成需求说明
- 把功能拆成可检查任务
- 生成实现计划和测试计划
- 维护中大型项目的一致性

如果你只是临时画一张图，Kiro 可能偏重；如果你要长期维护一个网站、分析平台或工作流，它的规格驱动思路很有价值。

6. API 中转站和模型聚合服务

很多用户会接触到“中转站”“转发站”或“模型聚合服务”。它们通常提供一个统一 API，把请求转发到不同模型供应商。

优点：

- 一个接口访问多个模型

- 支付和额度管理可能更方便
- 有些服务提供兼容 OpenAI 格式的接口

风险：

- 数据会经过第三方服务
- 稳定性和响应速度不可控
- 模型版本、价格和上下文长度可能变化
- 有些服务可能不清楚数据保留策略
- 可能不适合处理未公开论文、临床数据、密钥或商业代码

建议：

- 能用官方 API 或企业账号时，优先用官方渠道
- 不把真实患者数据、访问密钥、服务器密码交给不明中转服务
- 如果必须使用中转，先做脱敏和小样本测试
- 在项目文档中记录模型、供应商、日期和关键参数

生信分析中的安全边界

AI 工具很擅长写代码，但不理解你的真实实验约束。下面这些事情必须由人来确认：

- 样本分组是否正确
- 统计检验是否匹配实验设计
- 批次效应是否需要处理
- marker gene 是否符合生物学背景
- 过滤阈值是否合理
- 可视化是否夸大结论
- 结果是否可重复

对于涉及人类样本、临床数据和未公开项目的数据，先默认不能上传到外部 AI 服务。至少要做：

- 去除姓名、编号、地址等直接标识符
- 去除可回溯到个体的元数据
- 不上传原始 FASTQ、BAM、表达矩阵全量数据
- 只提供最小可复现示例
- 使用本地模型、企业合规服务或脱敏后的样例数据

推荐工作流

学习阶段

用 AI 做解释器，而不是代写器：

> 用初学者能理解的方式解释这段 `Seurat` 代码。请逐行说明输入、输出和关键参数。

> 我不理解 `NormalizeData`、`FindVariableFeatures`、`ScaleData` 的区别。请结合单细胞表达矩阵解释。

分析阶段

让 AI 帮你生成草稿，但保留人工审查：

根据这个数据框结构，写一段 ggplot2 代码画不同细胞类型的基因表达小提琴图。请不要假设不存在的列名。

下面是报错信息和 sessionInfo。请判断最可能的原因，先给排查步骤，不要直接改代码。

项目维护阶段

让 AI 做重复劳动：

检查 docs/basics 目录下的教程，找出标题层级不一致、图片路径可能错误、代码块语言未标注的问题。

为这个分析脚本生成 README，说明输入文件、输出文件、依赖包和运行命令。

提示词模板

解释代码

请解释下面这段代码。要求：
1. 说明每一步的目的
2. 标出输入和输出
3. 指出可能需要根据数据修改的参数
4. 不要重写代码，除非发现明确错误

生成分析脚本

请写一个可复现的 R 脚本完成以下任务：
- 输入：counts.csv 和 metadata.csv
- 输出：QC 图、标准化后的对象、marker 基因表
- 要求：固定随机种子，记录 sessionInfo，所有输出写入 results/
- 不要使用不存在的列名；如果需要列名，请先向我确认

审查结果

请作为代码审查者检查这段分析流程：
1. 是否有统计学问题
2. 是否有不可复现的步骤
3. 是否有硬编码路径
4. 是否遗漏中间结果保存
5. 是否需要补充图注或方法说明

工具选择建议

场景	更适合的工具
解释一段代码	ChatGPT、Claude、IDE 助手
修改本地项目多个文件	Codex、Claude Code、opencode
快速原型	Codex、Claude Code、Cursor、Kiro
规范化长期项目	Kiro、Codex、Claude Code
多模型切换	opencode、模型聚合服务
敏感数据分析	本地环境、脱敏数据、企业合规服务

下一步

建立了 AI 使用姿态，接着补上动手能力：

- 编程基础：R、Python、Bash 学习路径
- Jupyter 与交互式分析环境

参考资料

- OpenAI Codex CLI：https://developers.openai.com/codex/cli
- OpenAI Codex 使用说明：https://help.openai.com/en/articles/11369540/
- Claude Code 文档：https://code.claude.com/docs
- opencode 文档：https://dev.opencode.ai/docs
- Kiro 文档：https://kiro.dev/docs/
- Kiro 规格驱动开发介绍：https://kiro.dev/blog/introducing-kiro/

03 编程基础：R、Python、Bash 学习路径

做组学数据分析，编程不是目标，而是工具。你不需要一开始就成为程序员，但必须能读懂脚本、修改参数、运行流程、定位报错，并把结果整理成可重复的分析记录。

本章给出 BioF3 推荐的最小学习路径：先建立 R / Python / Bash 的基础能力，再用 AI 工具提高学习和开发效率。

为什么必须学编程

组学数据通常有三个特点：

- 数据量大：表达矩阵、FASTQ、BAM、H5AD、RDS 文件都不适合手工处理
- 步骤多：质控、标准化、降维、聚类、差异分析、富集分析都需要参数记录
- 结果要可重复：论文、报告和协作都要求别人能复现你的流程

如果只靠点鼠标，很难回答下面这些问题：

- 这次分析用了哪些过滤阈值?
- 归一化和聚类参数是什么?
- 哪些图来自哪一步脚本?
- 换一批样本能否自动重跑?
- 几个月后还能否复现同样结果?

编程的价值在于把分析过程写下来，让它可以检查、重复、修改和共享。

BioF3 推荐工具栈

R：单细胞分析和统计可视化主力

优先学习 R，因为 BioF3 的许多单细胞实践会用到 R 生态。

常见场景：

- Seurat 单细胞分析
- ggplot2 可视化
- 差异表达分析
- 统计检验和建模
- 富集分析
- R Markdown / Quarto 报告

最低要求：

- 会读写 CSV、TSV、RDS
- 会使用 data.frame / tibble
- 会用 dplyr 做筛选、分组、排序和汇总
- 会用 ggplot2 画常见图
- 会安装和加载包
- 能看懂函数参数和报错信息

Python：数据处理、机器学习和 Scanpy 生态

Python 适合处理大规模数据、机器学习和工程化流程。

常见场景：

- pandas / numpy 数据处理
- Scanpy / AnnData 单细胞分析
- scikit-learn 机器学习
- PyTorch / 深度学习
- 自动化脚本和 API 调用
- 文件批处理

最低要求：

- 会创建虚拟环境
- 会读写 CSV、TSV、JSON、H5AD
- 会使用 pandas 做表格处理
- 会使用 matplotlib / seaborn 画图
- 会写简单函数
- 能根据 traceback 定位错误

Bash：服务器和生信流程的入口

很多生信工具首先是命令行工具。不会 Bash，就很难稳定使用服务器和高通量流程。

常见场景：

- 查看和移动文件
- 解压和统计文件
- 批量运行 FastQC、Cell Ranger、STAR、samtools
- 后台任务和日志检查
- 远程服务器操作
- 调用脚本和管道命令

最低要求：

- 会 `cd` 、 `ls` 、 `cp` 、 `mv` 、 `mkdir` 、 `rm`
- 会 `head` 、 `tail` 、 `less` 、 `wc`
- 会 `grep` 、 `awk` 、 `sed` 的基本用法
- 会写简单 `for` 循环
- 会重定向输出和查看日志
- 会用 `ssh` 登录服务器

AI：助教、解释器和开发助手

AI 工具可以帮助你更快进入状态，但不要把它当成分析负责人。

适合让 AI 做：

- 解释陌生代码
- 把报错翻译成排查步骤
- 生成脚本草稿
- 改写重复代码
- 生成 README 或方法说明
- 检查路径、变量名、代码风格

不适合完全交给 AI 的事情：

- 决定实验分组
- 决定统计检验是否合理

- 判断 marker gene 是否可信
- 解释疾病机制
- 处理未脱敏的人类样本数据
- 编造软件版本和文献依据

更完整的工具介绍见：AI 辅助编程与智能体工具。

学习顺序

第 1 步：先学会运行和修改现有代码

新手最容易陷入“先系统学完整门语言”的误区。对组学分析来说，更有效的方式是先跑通已有脚本。

目标：

- 能打开 RStudio、Jupyter 或终端
- 能运行一段教程代码
- 能修改输入文件路径
- 能修改过滤阈值
- 能保存结果图和结果表

练习：

> 运行一个教程脚本，把输入文件路径改成自己的目录，把输出目录改成 `results/`。

第 2 步：R 入门到可用

先掌握下面这些能力。

```r
# 变量和向量
genes <- c("TP53", "BRCA1", "EGFR")
expr <- c(5.2, 3.8, 7.1)

# 数据框
gene_data <- data.frame(
  gene = genes,
  expression = expr
)

# 查看数据
head(gene_data)
str(gene_data)
summary(gene_data)

# 筛选
high_expr <- gene_data[gene_data$expression > 5, ]
```

再学习 dplyr 风格的数据处理。

```r
library(dplyr)

gene_data <- gene_data %>%
  mutate(log_expression = log2(expression + 1)) %>%
  arrange(desc(expression))
```

最后学习 ggplot2。

```r
library(ggplot2)

ggplot(gene_data, aes(x = gene, y = expression)) +
  geom_col(fill = "#3B82F6") +
  labs(x = "Gene", y = "Expression") +
  theme_classic()
```

R 阶段的 BioF3 目标不是语法完美，而是能读懂 Seurat 教程里的对象、函数和参数。

第 3 步：Python 入门到可用

Python 先从 pandas 开始。

```python
import pandas as pd
import numpy as np

data = pd.DataFrame({
    "gene": ["TP53", "BRCA1", "EGFR"],
    "expression": [5.2, 3.8, 7.1],
})

data["log_expression"] = np.log2(data["expression"] + 1)
high_expr = data[data["expression"] > 5]
summary = data["expression"].mean()

print(high_expr)
print(summary)
```

画图先掌握 matplotlib 和 seaborn。

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(data=data, x="gene", y="expression", color="#3B82F6")
plt.xlabel("Gene")
plt.ylabel("Expression")
plt.tight_layout()
plt.show()
```

Python 阶段的 BioF3 目标是能处理表格数据、理解 Scanpy / AnnData 的基本对象，并能写小型自动化脚本。

第 4 步：Bash 入门到可用

先掌握文件和目录。

```
pwd
ls -lh
mkdir -p results/qc
cp data/sample.csv results/
```

再掌握查看文件。

```
head expression.tsv
tail -n 20 run.log
wc -l expression.tsv
less run.log
```

然后学习批量处理。

```
mkdir -p qc_results

for file in data/*.fastq.gz; do
  echo "Processing $file"
  fastqc "$file" -o qc_results/
done
```

最后学习远程服务器。

```
ssh aliyun
scp results/report.html aliyun:/opt/project/results/
```

Bash 阶段的 BioF3 目标是能在服务器上找到文件、运行工具、检查日志和批量处理样本。

AI 辅助学习

用 AI 做解释器、排错助手、草稿生成器，可以显著提升学习效率。具体用法、提示词模板和安全边界详见 AI 辅助编程与智能体工具。

一个最小可复现项目结构

建议从一开始就把分析项目整理成固定结构。

```
project/
├── data/
│   ├── raw/
│   └── processed/
├── scripts/
│   ├── 01_qc.R
│   ├── 02_normalize.R
│   └── 03_plot.R
├── results/
│   ├── figures/
│   └── tables/
├── logs/
└── README.md
```

基本原则：

- 原始数据放 `data/raw/`，不要随意覆盖
- 中间数据放 `data/processed/`
- 脚本放 `scripts/`
- 图和表分别放 `results/figures/` 、 `results/tables/`
- 运行日志放 `logs/`
- README 记录运行顺序、软件版本和关键参数

8 周学习路线

时间	目标	练习
第 1 周	会运行 R / Python / Bash	运行教程代码，修改路径和输出目录
第 2 周	R 基础数据结构	读取表达表，筛选高表达基因
第 3 周	ggplot2 可视化	画柱状图、散点图、小提琴图
第 4 周	Python pandas	读取表格，分组统计，导出结果
第 5 周	Bash 文件处理	批量统计文件行数，检查日志
第 6 周	服务器基础	ssh 登录，上传下载文件，后台运行任务
第 7 周	AI 辅助调试	用 AI 解释报错并人工验证
第 8 周	小项目整合	完成一个从输入到图表输出的可复现分析

常见问题

没有编程基础，先学 R 还是 Python?

如果目标是尽快进入单细胞分析，先学 R。BioF3 的单细胞实践主要依赖 R / Seurat。Python 可以在后续学习 Scanpy、机器学习和自动化时补上。

Bash 一定要学吗?

要学最基础的。你不需要成为 Linux 专家，但要能在服务器上找到文件、运行命令、查看日志。否则很多上游流程会卡住。

可以全靠 AI 写代码吗?

不建议。AI 可以写草稿，但你必须确认输入数据、列名、统计方法、输出结果和生物学解释。尤其是涉及临床数据和未公开项目时，不要把原始数据直接上传给外部 AI 服务。

报错时应该怎么处理?

推荐顺序：

1. 复制完整报错，不只看最后一行
2. 确认对象是否存在、路径是否正确、包是否加载
3. 用小数据集复现问题
4. 搜索错误信息
5. 让 AI 根据代码、报错和环境信息给排查步骤
6. 修复后记录原因

学到什么程度可以开始跑单细胞教程?

能做到下面几件事就可以开始:

- 会运行 R 脚本
- 会安装和加载 R 包
- 会修改文件路径
- 会查看对象的基本信息
- 会保存图和表
- 会根据报错做基础排查

下一步

完成本章后,建议继续学习:

- Jupyter 与交互式分析环境
- 公共数据库与数据检索
- R 数据整理与 ggplot2 可视化
- 单细胞实践 01

编程能力不是背语法背出来的,而是在真实任务中练出来的。最好的开始方式是选一个小数据表,完成读取、筛选、绘图和保存结果这四件事。

04 Jupyter 与交互式分析环境

组学分析的第一个问题是：在哪里写下可以重复运行的分析过程?

Jupyter Notebook 把代码、说明、图表和运行结果放在一起，适合探索性分析和教学演示。本章介绍如何使用本地 Jupyter、JupyterLab、Google Colab 和服务器 Notebook，并说明 Notebook 与普通脚本的分工。

学习目标

完成本章后，你应该能够：

- 理解 Notebook 在可重复分析中的作用
- 区分本地 Jupyter、JupyterLab、Google Colab 和服务器环境
- 掌握 Notebook 的基本结构
- 知道什么时候该把 Notebook 稳定下来改为脚本

Notebook 适合做什么

Jupyter Notebook 是一种把代码、说明文字、图表和运行结果放在一起的交互式文档。它适合探索性分析和教学演示。

适合：

- 记录分析思路
- 逐步运行代码
- 快速查看表格和图
- 写下参数解释
- 分享小型分析示例

不适合：

- 长时间无人值守的大流程
- 大规模批处理任务
- 保存敏感数据输出
- 没有版本控制的正式生产流程

推荐做法：Notebook 用来探索和记录，稳定后把关键步骤整理成 `.R` 、 `.py` 或工作流脚本。

常见交互式环境

Jupyter Notebook

经典 Notebook 界面，适合初学者。文件通常是 `.ipynb` ，内部保存代码单元、Markdown 文本、输出结果和元数据。

启动方式：

```
jupyter notebook
```

JupyterLab

JupyterLab 是更完整的工作界面，支持 Notebook、终端、文本编辑器和文件浏览器。做真实项目时更推荐 JupyterLab。

启动方式：

```
jupyter lab
```

Google Colab

Google Colab 是云端 Notebook 环境，打开浏览器即可运行 Python。它适合教学和轻量实验，不适合长期保存重要环境或处理隐私数据。

适合：

- 快速试代码
- 课堂演示
- 共享小型 Notebook
- 临时使用 GPU

注意：

- 运行环境可能变化
- 会话可能断开
- 文件需要显式保存
- 不要上传未脱敏的人类样本数据

服务器 Notebook

在真实组学项目中，数据常放在服务器上。可以在服务器启动 JupyterLab，再通过浏览器访问。

一般流程：

```
ssh aliyun
cd /path/to/project
jupyter lab --no-browser --port 8888
```

如果需要从本地浏览器访问远程 Notebook，通常会用 SSH 端口转发：

```
ssh -L 8888:localhost:8888 aliyun
```

服务器环境要特别注意权限、数据位置和端口安全。

Notebook 的基本结构

Markdown 单元

用来写说明、记录参数和解释结果。

```
## 质量控制

本步骤过滤低质量细胞：

- `nFeature_RNA < 200`
- `percent.mt > 20`
```

Code 单元

用来运行代码。

```
import pandas as pd

metadata = pd.read_csv("data/metadata.csv")
metadata.head()
```

输出结果

输出可以是表格、图像、日志或错误信息。正式分析时，不建议只依赖 Notebook 里的输出；重要结果应该保存到 `results/` 。

```
metadata.to_csv("results/metadata_checked.csv", index=False)
```

记录运行环境

一个能重跑的 Notebook 至少要把运行环境记下来。

Python 中可以记录版本：

```
import sys
import pandas as pd

print(sys.version)
print(pd.__version__)
```

R 中可以记录环境：

```
sessionInfo()
```

这些信息对后续复现结果、排查问题都有用。

从 Notebook 走向脚本

Notebook 的价值不是"把所有分析塞在一个文件里"，而是帮你逐步检查数据结构、记录思路。稳定下来的步骤应该整理到 `.R` 或 `.py` 脚本，便于批量运行、版本控制和团队协作。

结构建议：

- `notebooks/` 放探索性分析和教学 Notebook
- `scripts/` 放稳定的可复现脚本
- `results/` 放最终图表、表格和报告

下一步

继续学习：

- 公共数据库与数据检索
- R 数据整理与 ggplot2 可视化
- 单细胞实践 01

参考资源

- Jupyter 官方文档：https://docs.jupyter.org/
- JupyterLab 文档：https://jupyterlab.readthedocs.io/
- Google Colab：https://colab.research.google.com/

05 公共数据库与数据检索

组学分析的第二个问题是：从哪里找到可信的数据，并确认它是否适合自己的问题？

本章介绍常用公共数据库 GEO、SRA、CELLxGENE、Human Cell Atlas 和 Expression Atlas 分别适合什么数据，以及如何根据 accession 编号追踪数据来源。

AI 工具如何辅助数据检索，详见 AI 辅助编程与智能体工具。

学习目标

完成本章后，你应该能够：

- 知道不同数据库分别适合解决什么问题
- 看懂常见 accession 编号的层级
- 用合适的关键词检索公共数据
- 判断一个数据集是否适合自己的研究问题

公共数据库怎么选

不同数据库解决不同问题。先判断你需要的是原始测序数据、处理后的表达矩阵，还是可交互浏览的注释数据。

需求	优先看哪里
找论文配套表达矩阵	GEO
下载原始 FASTQ	SRA / ENA
浏览单细胞注释数据	CELLxGENE
找人类细胞图谱项目数据	HCA Data Portal
查基因在组织/细胞中的表达	Expression Atlas

常见 accession 编号

公开数据通常通过 accession 编号追踪。

编号	常见含义	示例
GSE	GEO Series，一个研究或数据集	GSE12345
GSM	GEO Sample，一个样本	GSM123456
SRP	SRA Study	SRP123456
SRS	SRA Sample	SRS123456
SRX	SRA Experiment	SRX123456
SRR	SRA Run，常用于下载 reads	SRR1234567

分析前要确认编号层级。很多新手拿到 `GSE` 后直接找 FASTQ，会发现真正下载 reads 需要进一步找到对应的 `SRR`。

主要数据库

GEO

GEO 是 NCBI 维护的功能基因组学数据库，常见于论文数据提交。它可以包含表达矩阵、样本信息、平台信息和补充文件。

网址：https://www.ncbi.nlm.nih.gov/geo/

适合：

- 查找论文配套数据
- 下载处理后的表达矩阵
- 查看样本分组和元数据
- 追踪到 SRA 原始数据

检索建议：

```
关键词 + 物种 + 技术 + 组织/疾病
```

例如：

```
single cell RNA-seq human liver fibrosis
```

SRA

SRA 是 NCBI 的原始测序数据归档库。需要 FASTQ 时通常会用到它。

网址：https://www.ncbi.nlm.nih.gov/sra/

下载常用 SRA Toolkit：

```
conda install -c bioconda sra-tools

prefetch SRR1234567
fasterq-dump SRR1234567 --split-files -O data/raw/
```

注意：

- FASTQ 文件可能很大
- 下载前确认磁盘空间
- 批量下载前先测试一个 run
- 记录 SRR 列表和下载日期

CELLxGENE

CELLxGENE Discover 提供许多可浏览的单细胞数据集，通常可以在线查看 UMAP、细胞类型和基因表达。

网址：https://cellxgene.cziscience.com/

适合：

- 快速浏览单细胞数据
- 查看细胞类型注释
- 寻找可下载的 h5ad 数据
- 比较公开数据中的基因表达模式

使用建议：

- 先在线检查数据是否符合研究问题
- 下载前确认样本、组织、物种和处理流程
- 注意数据是否已经标准化或整合

Human Cell Atlas

Human Cell Atlas 关注人类细胞参考图谱。HCA Data Portal 提供社区生成的多组学开放数据。

网址：https://data.humancellatlas.org/

适合：

- 查找人类组织和器官图谱
- 获取大型参考数据
- 了解细胞类型注释和 atlas 项目
- 作为数据整合和注释参考

Expression Atlas / Single Cell Expression Atlas

EMBL-EBI 的 Expression Atlas 和 Single Cell Expression Atlas 提供基因表达查询和单细胞表达浏览。

网址：https://www.ebi.ac.uk/gxa/

适合：

- 查询某个基因在不同组织或细胞中的表达
- 浏览经过整理的表达数据
- 做初步假设生成
- 辅助解释 marker gene

下一步

继续学习：

- R 数据整理与 ggplot2 可视化
- 单细胞实践 01：实践数据集与数据获取

参考资源

- GEO：https://www.ncbi.nlm.nih.gov/geo/
- SRA：https://www.ncbi.nlm.nih.gov/sra/
- CELLxGENE：https://cellxgene.cziscience.com/
- Human Cell Atlas Data Portal：https://data.humancellatlas.org/
- Expression Atlas：https://www.ebi.ac.uk/gxa/

06 R 数据整理与 ggplot2 可视化

R 在组学分析中最常用的场景有两个：整理表格数据，画出能解释结果的图。对 BioF3 来说，本章不是追求完整覆盖 R 语言，而是让你掌握最常用、最容易迁移到真实项目的能力。

本章所有图表都来自公开数据集 10x Genomics PBMC 3k。它包含一名健康供体外周血单个核细胞的真实单细胞 RNA-seq 表达矩阵。

学习目标

完成本章后，你应该能够：

- 理解 R 中向量、数据框和列表的基本用途
- 用 dplyr 完成筛选、排序、分组和汇总
- 理解 ggplot2 的图形语法
- 选择适合生信问题的图表类型
- 保存可复现的高质量图片
- 用 AI 辅助改图，但能自己判断图是否合理

R 在 BioF3 中承担什么角色

R 不是唯一选择，但它在生信里非常重要：

- Seurat 是单细胞分析的主流工具之一
- Bioconductor 提供大量组学分析包
- ggplot2 和 ComplexHeatmap 适合发表级可视化
- R Markdown / Quarto 适合生成分析报告

学习 R 的重点不是背语法，而是知道数据对象长什么样、函数需要什么输入、结果如何保存。

准备真实 PBMC 3k 数据

先读取真实表达矩阵。完整脚本会自动下载数据；这里保留核心逻辑，方便你理解后续对象从哪里来。

```
library(Matrix)
library(dplyr)
library(tidyr)
library(ggplot2)
library(scales)

data_dir <- file.path(path.expand("~"), "biof3-data", "pbmc3k")
matrix_dir <- file.path(data_dir, "filtered_gene_bc_matrices", "hg19")

pbmc_url <- paste0(
  "https://cf.10xgenomics.com/samples/cell-exp/1.1.0/pbmc3k/",
  "pbmc3k_filtered_gene_bc_matrices.tar.gz"
)
pbmc_tar <- file.path(data_dir, "pbmc3k_filtered_gene_bc_matrices.tar.gz")

dir.create(data_dir, recursive = TRUE, showWarnings = FALSE)

if (!file.exists(pbmc_tar)) {
  download.file(pbmc_url, destfile = pbmc_tar, mode = "wb")
}

if (!file.exists(file.path(matrix_dir, "matrix.mtx"))) {
  untar(pbmc_tar, exdir = data_dir)
}

counts <- readMM(file.path(matrix_dir, "matrix.mtx"))
genes <- read.delim(file.path(matrix_dir, "genes.tsv"), header = FALSE)
barcodes <- read.delim(file.path(matrix_dir, "barcodes.tsv"), header = FALSE)

rownames(counts) <- make.unique(genes[[2]])
colnames(counts) <- barcodes[[1]]
counts <- as(counts, "CsparseMatrix")
```

把矩阵整理成几张常用表：

```r
mt_genes <- grepl("^MT-", rownames(counts))

qc <- data.frame(
  cell = colnames(counts),
  nCount_RNA = as.numeric(colSums(counts)),
  nFeature_RNA = as.numeric(colSums(counts > 0)),
  percent_mt = as.numeric(colSums(counts[mt_genes, , drop = FALSE]) / colSums(counts) * 100)
)

gene_summary <- data.frame(
  gene = rownames(counts),
  total_counts = as.numeric(rowSums(counts)),
  detected_cells = as.numeric(rowSums(counts > 0))
) %>%
  mutate(mean_counts = total_counts / ncol(counts))

marker_genes <- c("IL7R", "CCR7", "S100A8", "S100A9", "MS4A1", "CD79A", "NKG7", "GNLY", "PPBP")
marker_genes <- marker_genes[marker_genes %in% rownames(counts)]

marker_long <- bind_rows(lapply(marker_genes, function(gene) {
  data.frame(
    cell = colnames(counts),
    gene = gene,
    counts = as.numeric(counts[gene, ]),
    log_counts = log1p(as.numeric(counts[gene, ]))
  )
}))
```

这几张表分别回答不同问题：

- `counts`：基因 x 细胞的稀疏表达矩阵
- `qc`：每个细胞的总 UMI、检测基因数、线粒体比例
- `gene_summary`：每个基因的总表达量和检出细胞数
- `marker_long`：常见 PBMC marker 基因的长表表达量

最小 R 基础

向量

向量是 R 中最基础的数据结构。这里的基因名来自 PBMC 3k 矩阵中真实存在的 marker 基因。

```r
genes <- marker_genes[1:4]
expression <- gene_summary$total_counts[match(genes, gene_summary$gene)]

genes[1]
expression > median(expression)
mean(expression)
```

数据框

数据框类似表格，是生信分析中最常见的数据形态。

```r
gene_data <- gene_summary %>%
  filter(gene %in% marker_genes) %>%
  select(gene, total_counts, detected_cells, mean_counts)

head(gene_data)
str(gene_data)
summary(gene_data)
```

访问列：

```r
gene_data$gene
gene_data[["total_counts"]]
```

筛选行：

```r
high_detection <- gene_data[gene_data$detected_cells > 100, ]
```

列表

很多 R 包会把复杂结果放在列表里。Seurat 对象、差异分析结果和富集分析结果都可能包含多层信息。

```r
analysis_result <- list(
  counts = counts,
  qc = qc,
  marker_expression = marker_long
)

analysis_result$qc
```

dplyr：表格整理的基本动作

先加载包：

```r
library(dplyr)
```

用真实基因汇总表练习筛选、排序和新增列：

```r
top_genes <- gene_summary %>%
  filter(!grepl("^MT-|^RPL|^RPS", gene)) %>%
  filter(detected_cells > 50) %>%
  arrange(desc(total_counts)) %>%
  mutate(
    log_total_counts = log10(total_counts + 1),
    detection_rate = detected_cells / ncol(counts)
  ) %>%
  slice_head(n = 10)
```

分组汇总可以用在 marker 基因上。例如按基因统计非零表达细胞比例：

```r
marker_summary <- marker_long %>%
  group_by(gene) %>%
  summarise(
    mean_log_counts = mean(log_counts),
    expressing_cells = sum(counts > 0),
    expressing_rate = expressing_cells / n(),
    .groups = "drop"
  ) %>%
  arrange(desc(expressing_rate))
```

在真实项目中，大部分绘图问题都先是数据整理问题。图画不好，常常不是 ggplot2 不会用，而是表格没有整理成合适的长格式。

---

ggplot2 的核心思想

ggplot2 使用“图形语法”。你可以把一张图理解成几层：

- data：使用哪个数据框
- aes：哪些列映射到 x、y、颜色、形状、大小
- geom：用什么几何对象展示数据
- scale：坐标轴和颜色如何转换
- theme：图的外观
- labs：标题、坐标轴和图例文字

最小示例：

```r
ggplot(top_genes, aes(x = reorder(gene, total_counts), y = total_counts)) +
  geom_col(fill = "#0f766e") +
  coord_flip() +
  scale_y_continuous(labels = comma) +
  labs(x = NULL, y = "Total UMI counts") +
  theme_classic()
```

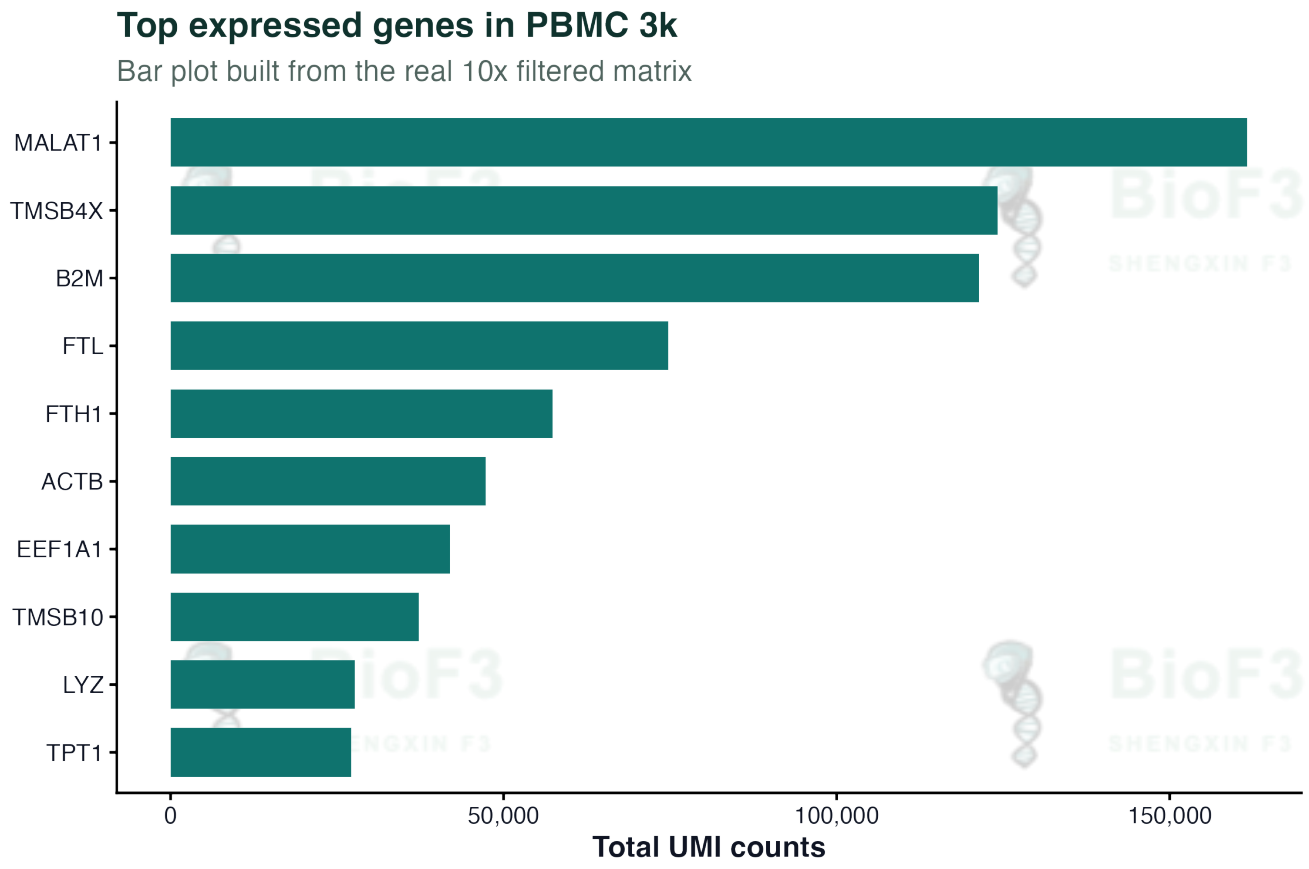

图 1：柱状图展示 PBMC 3k 中总 UMI 数最高的一组真实基因。这里过滤了线粒体和核糖体基因，避免它们占据整个图。

生信常用图表

散点图：看两个变量的关系

```
ggplot(qc, aes(x = nCount_RNA, y = nFeature_RNA, color = percent_mt)) +
  geom_point(alpha = 0.7, size = 1.2) +
  scale_x_continuous(labels = comma) +
  scale_y_continuous(labels = comma) +
  scale_color_gradient(low = "#0f766e", high = "#dc2626") +
  labs(
    x = "Total UMI counts",
    y = "Detected genes",
    color = "Mitochondrial %"
  ) +
  theme_classic()
```

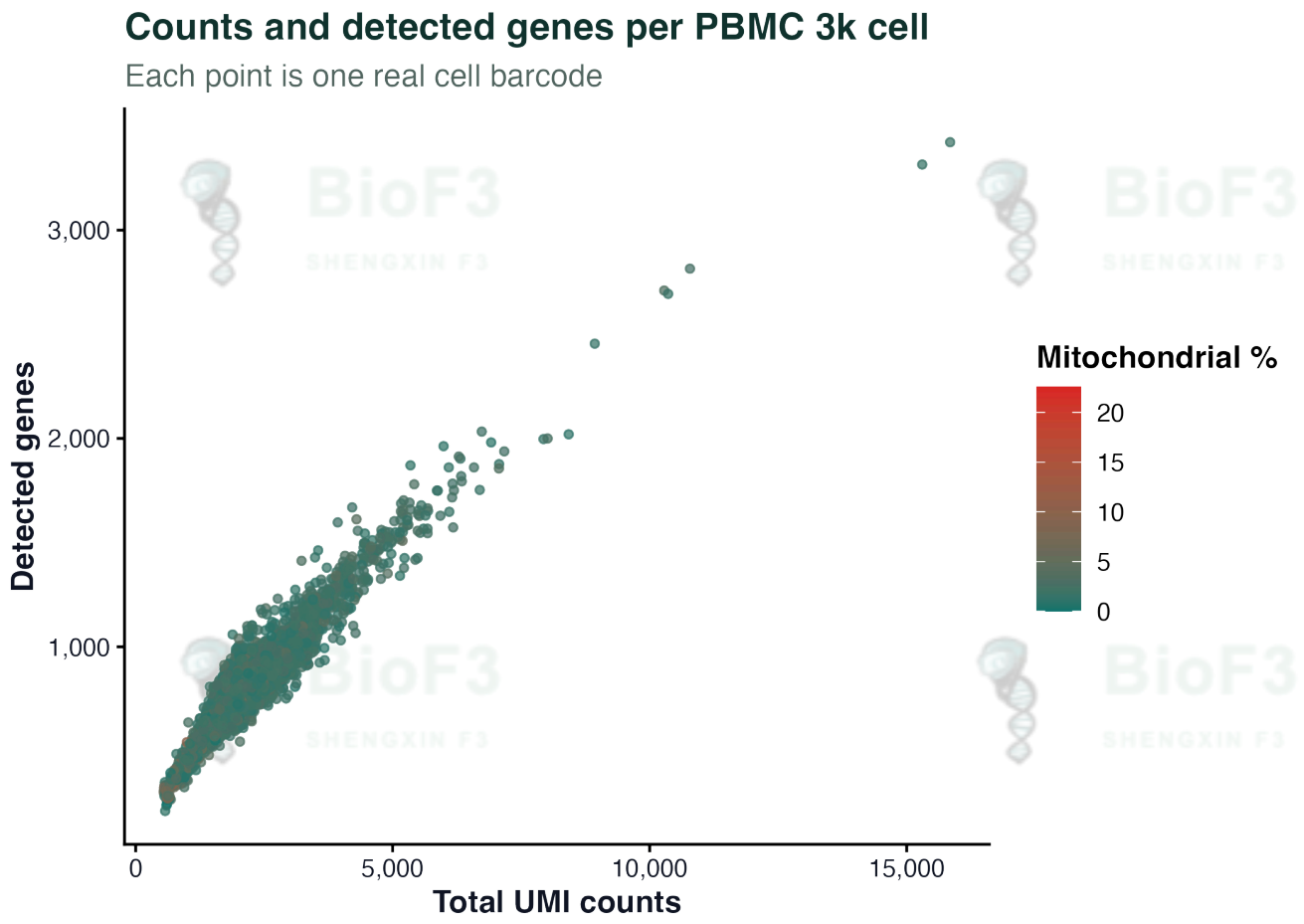

图 2：每个点是一个真实细胞条形码。散点图适合观察总 UMI 数、检测基因数和线粒体比例之间的关系。

箱线图：比较分组分布

```
box_data <- marker_long %>%
  filter(gene %in% marker_genes[1:4])

ggplot(box_data, aes(x = gene, y = log_counts, fill = gene)) +
  geom_boxplot(width = 0.6, outlier.shape = NA, alpha = 0.82) +
  labs(x = NULL, y = "log1p(UMI counts)") +
  theme_classic() +
  theme(legend.position = "none")
```

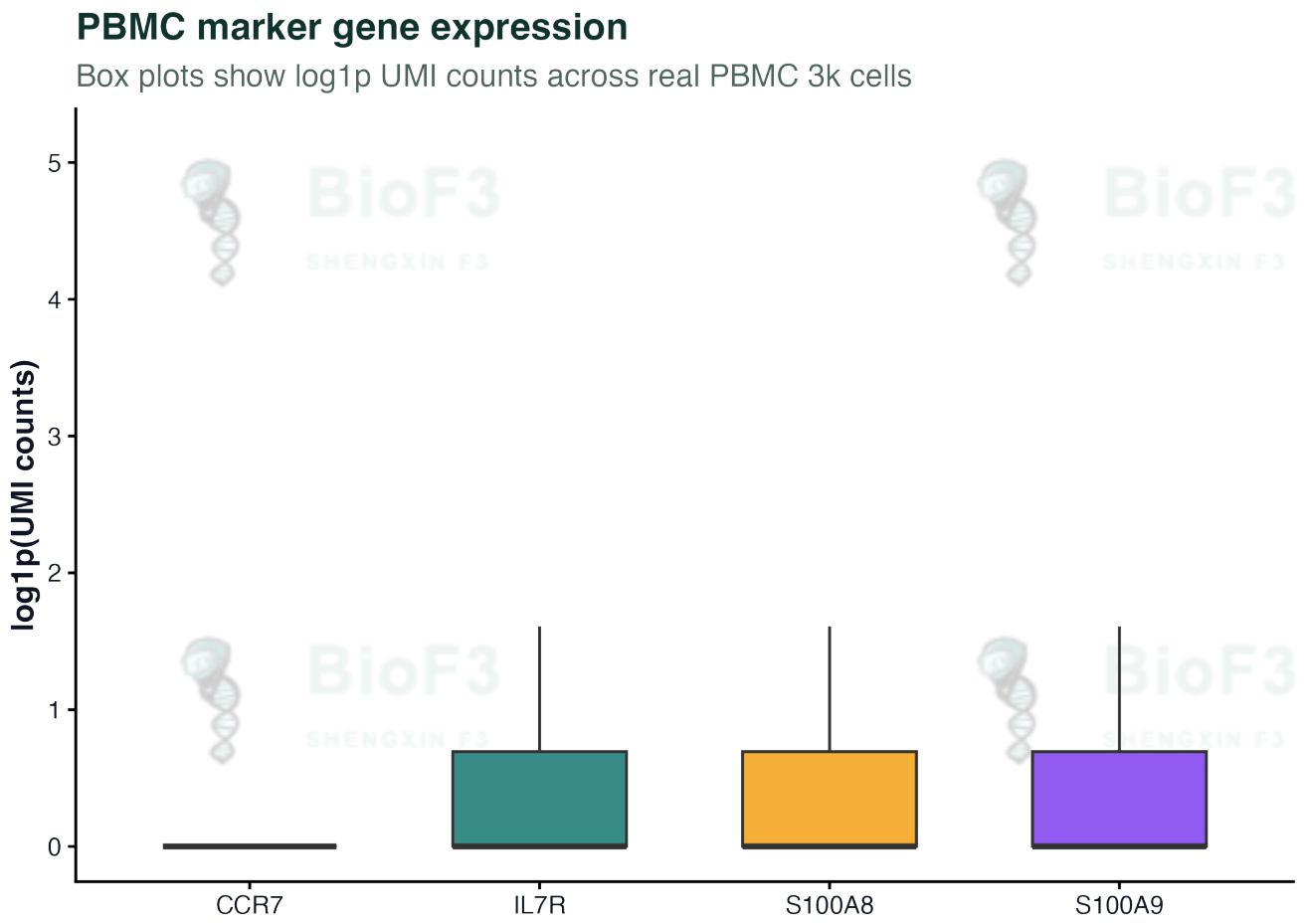

图 3：箱线图展示真实 PBMC marker 基因在全部细胞中的表达分布。单细胞数据里零值很多，所以图注必须说明归一化或转换方式。

直方图：看整体分布

```
ggplot(qc, aes(x = nCount_RNA)) +
  geom_histogram(bins = 55, fill = "#2563eb", color = "white") +
  geom_vline(xintercept = median(qc$nCount_RNA), linetype = "dashed", color = "#f59e0b") +
  scale_x_continuous(labels = comma) +
  labs(x = "Total UMI counts per cell", y = "Number of cells") +
  theme_classic()
```

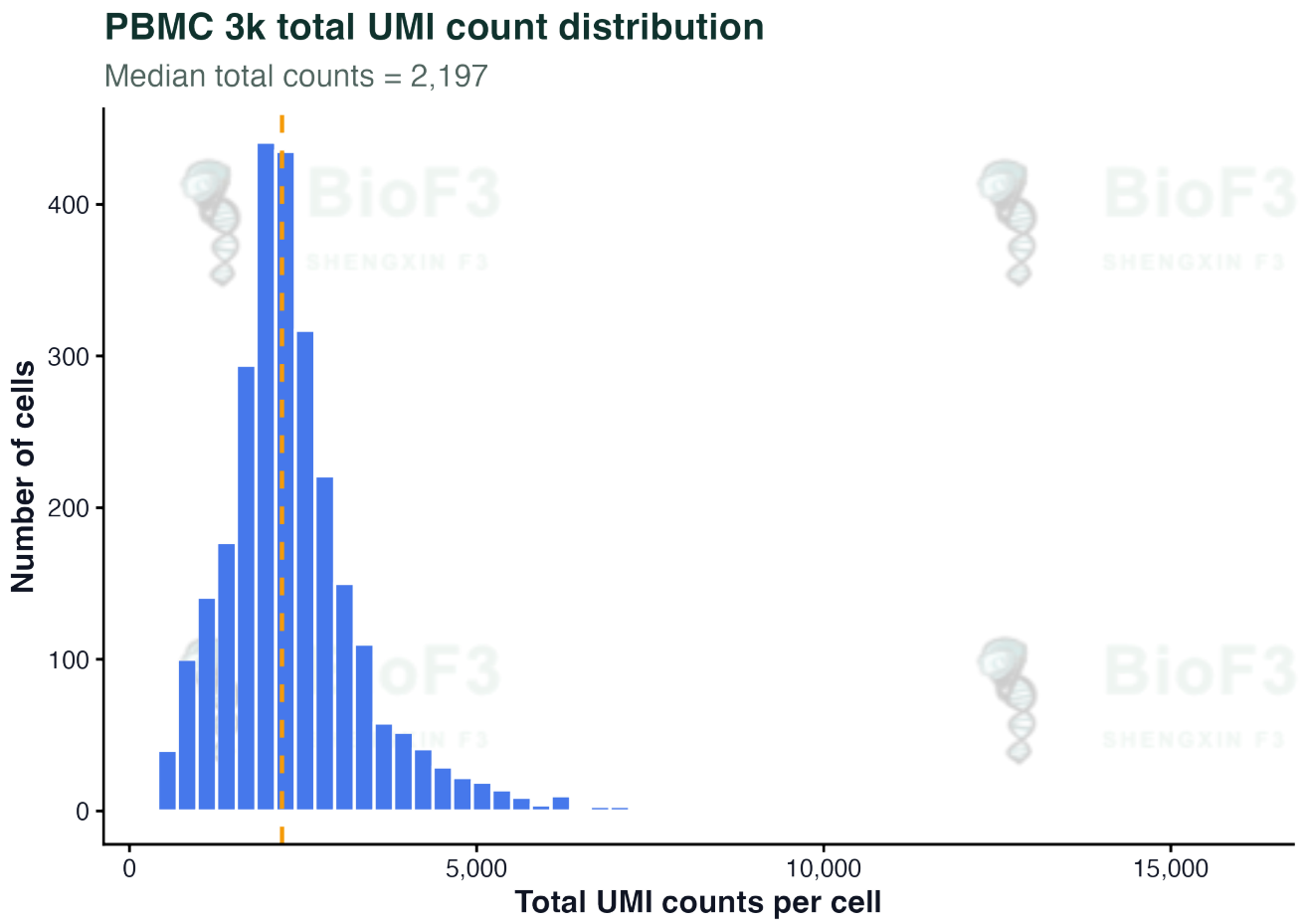

图 4：直方图适合观察真实测序计数、基因表达、QC 指标等连续变量的分布。

小提琴图：看分布形状

```
ggplot(box_data, aes(x = gene, y = log_counts, fill = gene)) +
  geom_violin(trim = FALSE, alpha = 0.72) +
  geom_boxplot(width = 0.12, fill = "white", outlier.shape = NA) +
  labs(x = NULL, y = "log1p(UMI counts)") +
  theme_classic() +
  theme(legend.position = "none")
```

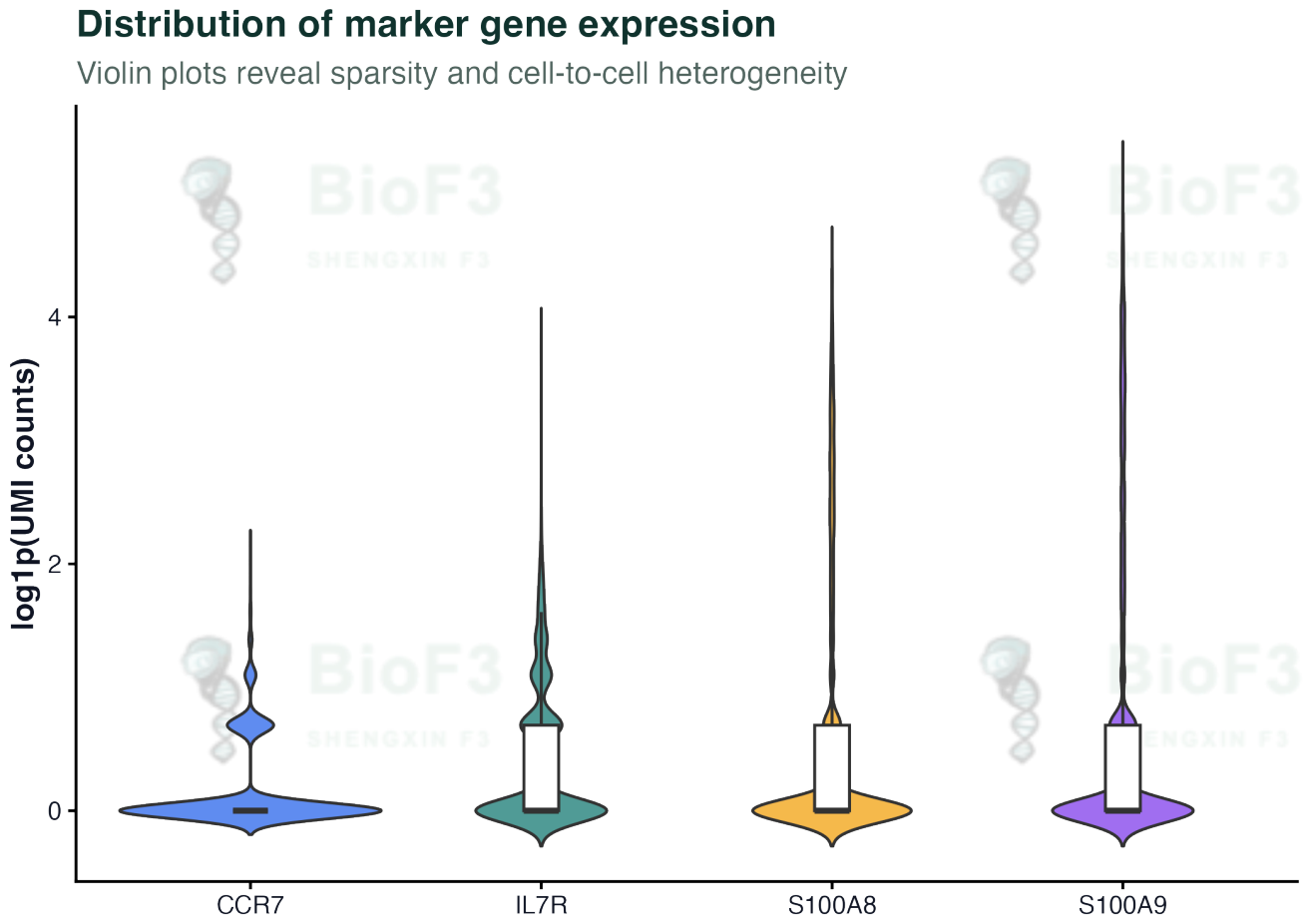

图 5：小提琴图适合展示大量细胞的分布形状。单细胞表达图常用小提琴图，但要注意零值、归一化方式和细胞类型混合带来的解释边界。

分面图：同时比较多个基因或分组

```
facet_data <- marker_long %>%
  filter(gene %in% marker_genes[1:6])

ggplot(facet_data, aes(x = log_counts)) +
  geom_histogram(bins = 35, fill = "#0f766e", color = "white") +
  facet_wrap(~ gene, scales = "free_y", ncol = 3) +
  labs(x = "log1p(UMI counts)", y = "Cells") +
  theme_bw()
```

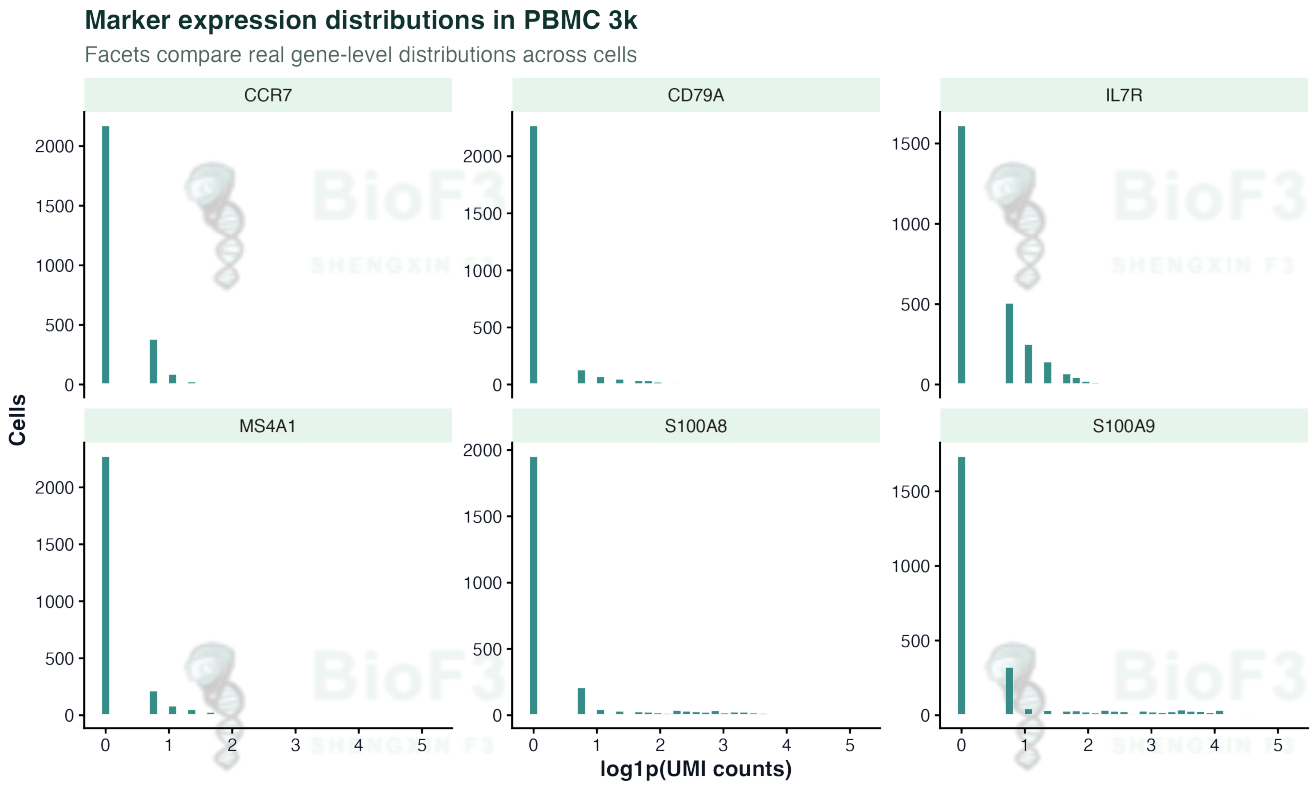

图 6：分面图适合在同一套样式下比较多个真实基因。这里每个小面板都是一个 marker 基因在 PBMC 3k 细胞中的表达分布。

一个小型表达分析流程

下面用 PBMC 3k 的真实 marker 基因演示从矩阵到长表、再到可视化的完整流程。

```
marker_expression <- counts[marker_genes, , drop = FALSE]

gene_long <- as.data.frame(as.matrix(marker_expression)) %>%
  tibble::rownames_to_column("gene") %>%
  pivot_longer(
    cols = -gene,
    names_to = "cell",
    values_to = "counts"
  ) %>%
  mutate(log_counts = log1p(counts))
```

可视化表达分布：

```
ggplot(gene_long, aes(x = gene, y = log_counts, fill = gene)) +
  geom_violin(trim = FALSE, alpha = 0.72) +
  geom_boxplot(width = 0.1, fill = "white", outlier.shape = NA) +
  labs(x = NULL, y = "log1p(UMI counts)") +
  theme_classic() +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 30, hjust = 1)
  )
```

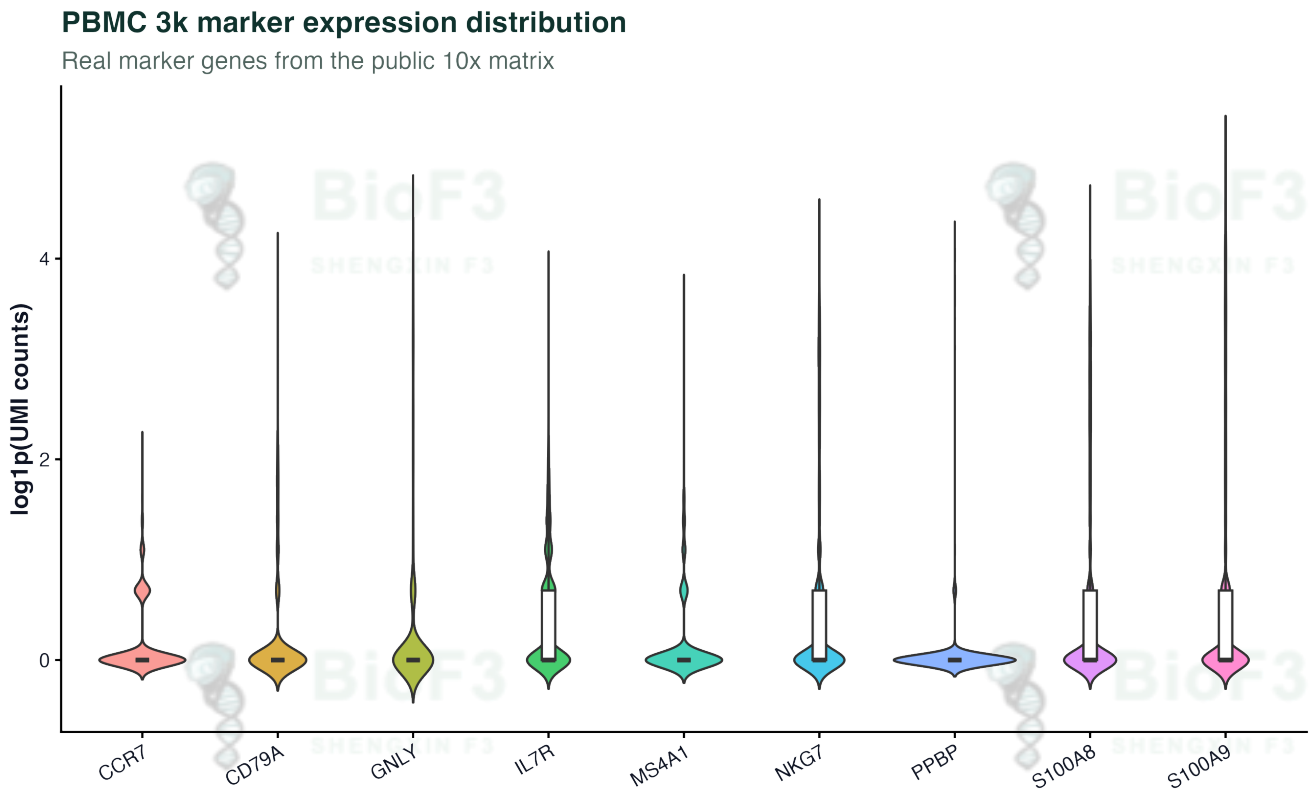

图 7：表达量分布图可以快速检查不同 marker 基因的稀疏性和表达范围。它不是差异分析结果，不能直接推断细胞类型比例或显著性。

基因检出、丰度和热图

火山图通常需要两类信息：

- x 轴：log2 fold change
- y 轴：显著性，例如 `-log10(adjusted p-value)`

PBMC 3k 这一章还没有建立分组差异分析模型，所以这里不画“正式火山图”。如果没有真实分组、真实 log2FC 和校正后的 p 值，就不要把图包装成火山图。我们改用真实基因层面的检出细胞数和总 UMI 数，观察哪些 marker 基因在矩阵中更常被检测到。

```
gene_detection <- gene_summary %>%
  filter(detected_cells > 0) %>%
  mutate(marker = if_else(gene %in% marker_genes, "Marker gene", "Other gene"))

ggplot(gene_detection, aes(x = detected_cells, y = total_counts, color = marker)) +
  geom_point(alpha = 0.45, size = 1) +
  scale_x_log10(labels = comma) +
  scale_y_log10(labels = comma) +
  labs(
    x = "Cells with detected expression",
    y = "Total UMI counts",
    color = NULL
  ) +
  theme_classic()
```

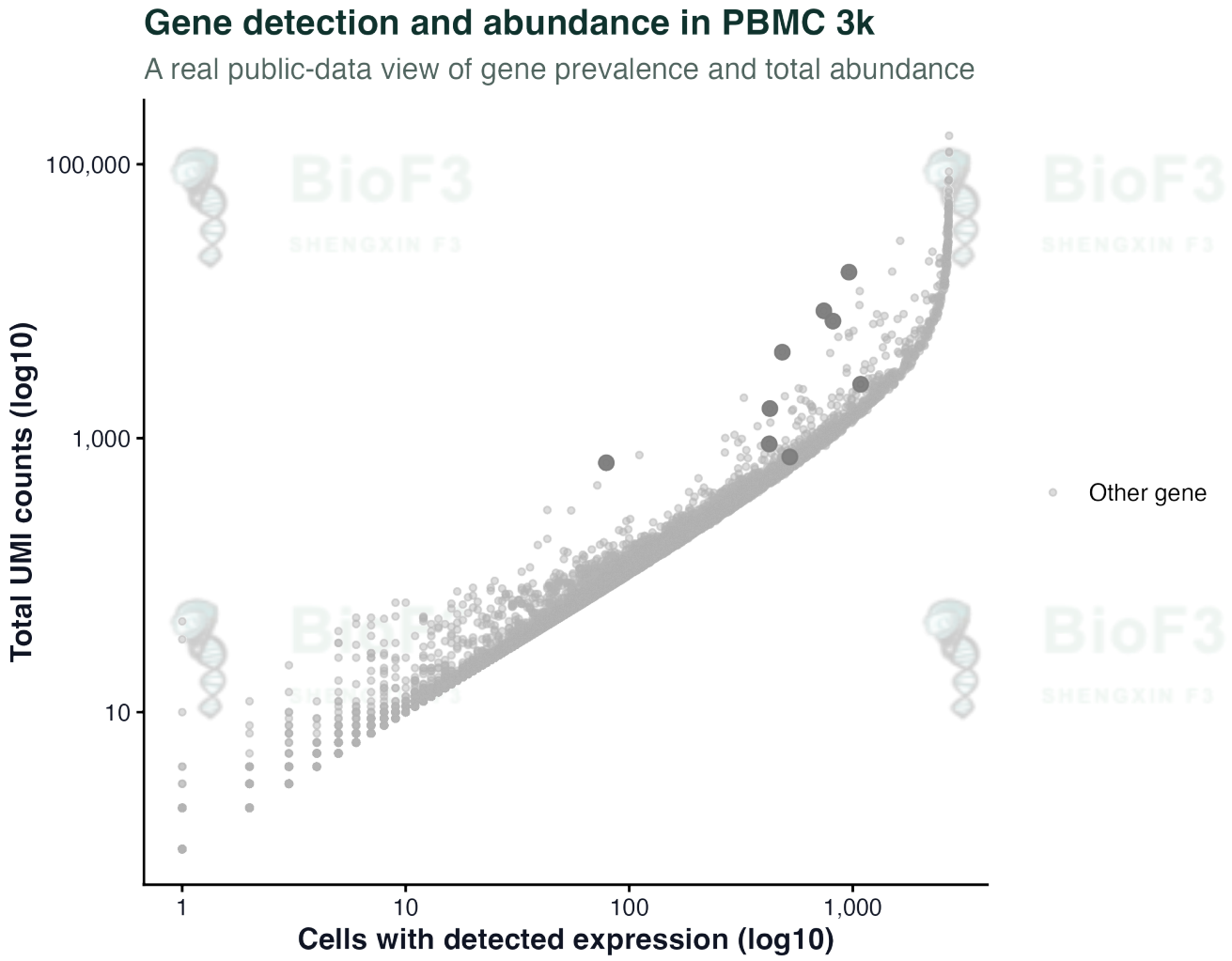

图 8：这不是火山图，而是 PBMC 3k 真实基因的“检出细胞数 vs 总 UMI 数”散点图。它保留了原文件名以兼容网站旧链接。

热图适合展示一组基因在多个样本或细胞中的模式。常见注意点：

- 是否按行标准化
- 是否显示聚类
- 是否展示样本分组注释
- 颜色是否有清晰含义
- 基因数量是否过多

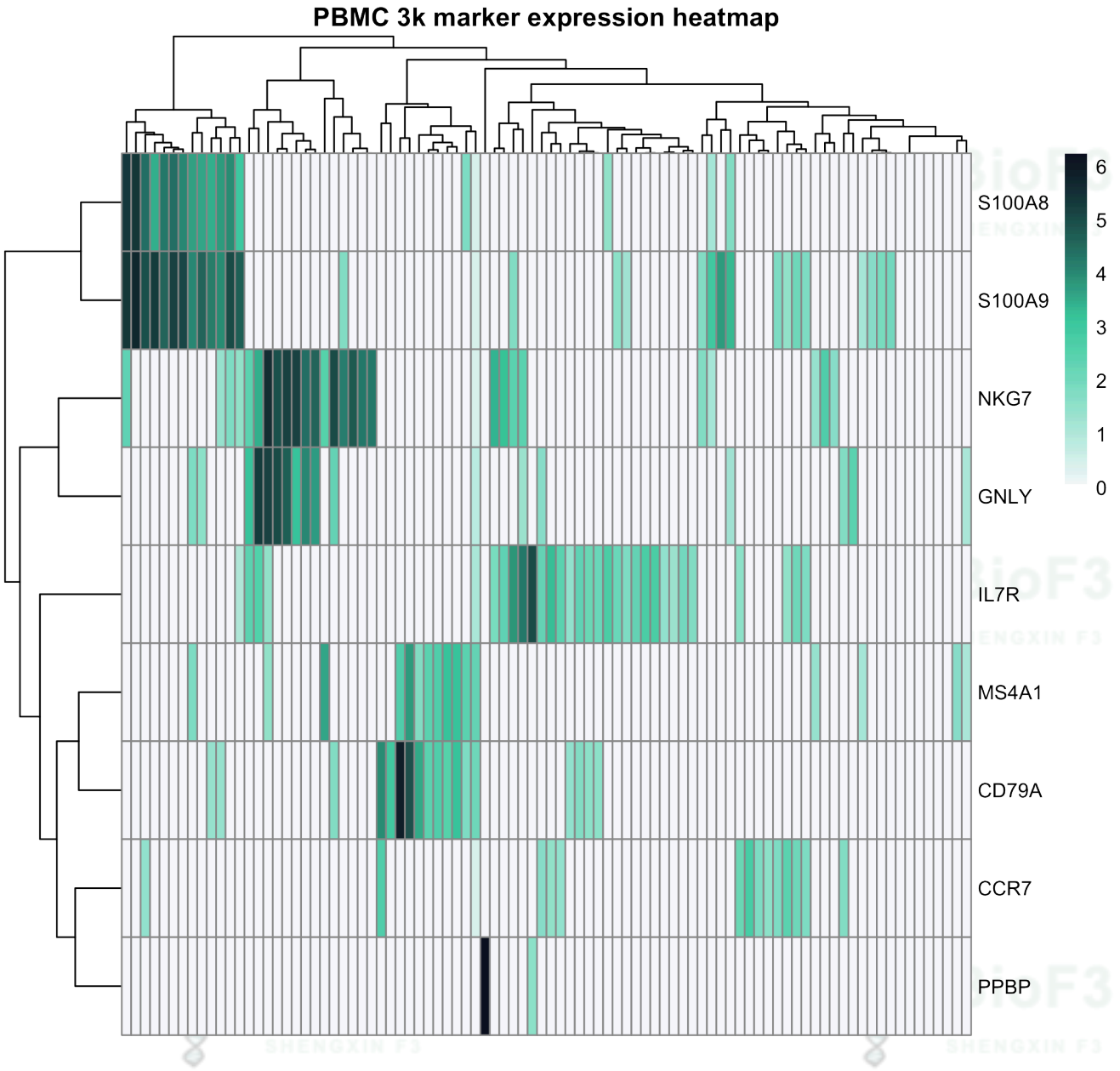

图 9：热图展示 PBMC 3k marker 基因在一组真实细胞中的表达模式。热图适合展示模式，而不是替代统计检验。

保存图片

建议所有图都显式保存，避免只留在 Notebook 或 RStudio 窗口里。

```
ggsave(
  filename = "results/figures/pbmc3k_marker_distribution.png",
  width = 7,
  height = 5,
  dpi = 300
)
```

保存前确认：

- 图片尺寸是否适合论文或网页
- 字体是否清晰
- 坐标轴是否有单位
- 图例是否能解释颜色

- 文件名是否能看出内容

AI 辅助改图

AI 可以帮你把"能画出来的图"改成"更清楚的图"：检查坐标轴、图例、颜色和图注，或者把宽表改成适合 ggplot2 的长表。具体提示词和安全边界详见 AI 辅助编程与智能体工具。

但下面这些事情必须自己判断：

- 用箱线图还是小提琴图是否符合数据量
- 是否应该展示每个细胞点，还是用分布图避免过度拥挤
- 是否需要多重检验校正
- 是否能从图上得出生物学结论
- 图注是否准确描述了数据处理方式

下载资源

module02_complete_sci.R
17 KB

下载图表生成脚本 ↗

下一步

基础入门到这里结束。继续学习：

- 单细胞实践 01：实践数据集与数据获取
- 单细胞实践 02：原始数据处理与 Cell Ranger

参考资源

- 10x Genomics PBMC 3k 数据集：https://www.10xgenomics.com/datasets/3-k-pbm-cs-from-a-healthy-donor-1-standard-1-1-0
- ggplot2 官方文档：https://ggplot2.tidyverse.org/
- dplyr 官方文档：https://dplyr.tidyverse.org/
- R for Data Science：https://r4ds.hadley.nz/
- Bioconductor：https://www.bioconductor.org/